Int j simul model 22 (2023) 2, 211-220
Original scientific paper

MODELLING AND SIMULATION PROCESS OF EXTENDED PETRI NETS WITH PNML AND MATLAB/SIMULINK

Alcaraz-Mejia, M.[#]; Parres-Peredo, A.; Piza-Davila, I. & Gutierrez-Preciado, L. F.
Electronics, Systems, and Informatics Department, ITESO University, Jalisco 45604, Mexico
E-Mail: mildreth@iteso.mx, parres@iteso.mx, hpiza@iteso.mx, lgutierrez@iteso.mx
([#] Corresponding author)

Abstract
In this work, we introduce a modelling and simulation process based on Matlab/Simulink for Petri nets that considers inhibitor arcs and priorities of transitions, herein named eXtended Petri nets, for modelling deterministic complex systems. The presented process comprises the method for modelling the system, the functions for the import to Matlab of all the data structures that define a Petri net with inhibitory arcs and priorities in transitions, and the functions to perform the simulation of the Petri net behaviour as a Simulink block. We present an exploratory case study about Rate Monotonic Scheduling for tasks with harmonic periods to show the complete modelling and simulation process. First, the scheduling system is modelled with Renew editor and saved as a PNML file. Then, this file is read and transformed into a Matlab data type. The produced data structures are the inputs to the proposed Simulink block, which performs the dynamic of the eXtended Petri net. Finally, the outputs of the simulation help validate the logical and temporal correctness of the scheduler model used as a case study. The concluding remarks section provides a link for downloading and testing the simulation process.
(Received in September 2022, accepted in February 2023. This paper was with the authors 2 weeks for 1 revision.)

Key Words: Discrete Event Systems, Petri Nets, Modelling, Simulation, Matlab, Simulink

1. INTRODUCTION

Petri nets (PN) are a widely used formalism, first introduced by Carl Petri [1], to model and validate a variety of systems from areas such as software, embedded systems, manufacturing, biological, chemical, mechanical, robotics, management, and many others. Informally, a PN is a bipartite graph composed of two types of nodes named places and transitions, representing events and conditions, respectively. The system state is represented by tokens, which reside in places and move between the places passing through transitions. Such dynamic is given by a state equation, which considers the structure of the net, i.e., the connection of places and transitions, as matrices. The mathematical representation and the graphical illustration provide the PN framework's main advantage.

In the state of the art, there exist diverse proposals with variants or extensions of PN, e.g., general Place/Transition Petri nets [2], Input-Output Petri nets [3], nets that consider time in transitions [4] or places [5], nets adapted to consider continuous time [6], hybrid nets combining discrete and continuous time [7], stochastic Petri nets [8], heuristic Petri nets [9], nets considering inhibitor arcs [10], nets with different types of priorities [11], and combination of these proposals [12]. These variants of PN help to cope with a wide range of systems problems such as controller design [13-15], system diagnosis [16, 17], reconfiguration systems [18, 19], and many others, either for logical or temporal validity.

To take advantage of the modelling throughout any variant of PN, companies, academia, and the community have proposed tools and software to model, simulate and analyse the system net. As Alcaraz-Mejia and Campos-Rodriguez mentioned [20], there are a variety of PN tools but only a few are based on Matlab/Simulink. Matlab/Simulink is a programming environment used for simulation in many areas such as for scheduling [21] and simulation [22]. Specifically, in PN area, Júlvez et al. [23] present a review of the following tools for simulation of Petri nets based on Matlab. Petri Net Toolbox and the Petri Net Simulink Block (PNSB), an extension for

Petri Net Toolbox, are both introduced in [24] and [25]. PN toolbox is one of the most complete tools based on Matlab and improved by the PNSB. This tool works with PN variants such as those with time in places or transitions, and with hybrid PN that considers events and continuous dynamics. In addition, it allows for the analysis of structural and behavioural properties, and computing of T- and P-semiflows. The PNSB extension allows for the simulation of the SimHPN simulator [26], and the HYbrid Petri Net Simulator (HYPENS) [27], but none of these tools works with inhibitor arcs and priorities simultaneously, and in the case of priorities, they only allow it as a probability of fire, which leads to a nondetermined behaviour.

In this work, we introduce a modelling and simulation process based on Matlab/Simulink for Petri nets that considers inhibitor arcs and priorities in transitions herein named eXtended Petri nets (XPN). For scientists and practitioners with experience in the use of Matlab, it may be simpler and more useful to import all the data structures that define a Petri net with inhibitory arcs and priorities in transitions, to use later the tools that Matlab provides. Additionally, with these structures imported into Matlab, their simulation using Simulink is facilitated. To the best of our knowledge, this is the first methodology providing some tools to ease the analysis of XPN (including inhibit and priority functions) based on Matlab/Simulink.

To sum up, the contributions of this work include the following: (1) a definition of XPN that consider prioritization on transitions besides inhibitor arcs, together with the induced dynamic, to model and simulate deterministic complex systems; (2) a Matlab function to transform XPN from PNML (Petri Net Marking Language) to Matlab data types; (3) a function to implement the dynamic of the XPN; (4) a Simulink Block for the simulation of the XPN; and, (5) a case study of a Rate Monotonic Scheduling for tasks with harmonic periods.

Then, the summary of the advantages of this work are:

- A type of PN to model deterministic complex systems.
- A guide for modelling XPN to save it as PNML.
- A function to import the XPN as data in Matlab, then we can use all the functioning provided by Matlab.
- A function that implements the dynamics of XPN.
- A basic Simulink block incorporating the dynamic of XPN, which works with the PN data in the Matlab workspace.

The rest of the paper is organized as follows. Section 2 introduces the definition of eXtended Petri nets and their dynamic. In section 3 a transformation of XPN from PNML to Matlab data types is given. Section 4 presents the m-functions to support the import of the PN from a PNML file. Section 5 presents the m-function to implement the dynamic from the eXtended Petri nets. Section 6 shows the simulation framework in Simulink and the implementation results. Finally, we provide our conclusions.

2. EXTENDED PETRI NETS

2.1 eXtended Petri net definition

The presented model uses an extension of Petri nets that consider a priority function [28] and an inhibitor function [29, 30]. Now, we introduce the basic notions for Petri net modelling.

The structure of the eXtended PN (XPN) is a directed graph $G = \{P, T, I, O, \phi, \rho\}$, where: $P = \{p_1, p_2, \ldots, p_m\}$ is a set of places with $|P| = m$, $T = \{t_1, t_2, \ldots, t_n\}$ is a set of transitions with $|T| = n$, $I: P \times T \rightarrow \mathrm{W}$ is an input function defining a flow relation from places to transitions, $O: P \times T \rightarrow \mathrm{W}$ is an output function defining a flow relation from transitions to places, $\phi: P \times T \rightarrow \{1, 0\}$ is an inhibitor function defining a flow blocking relation, and $\rho: T \rightarrow \mathrm{N}$ is a function defining the priority for the firing of transitions. N and W are the set of natural numbers and whole numbers, respectively.

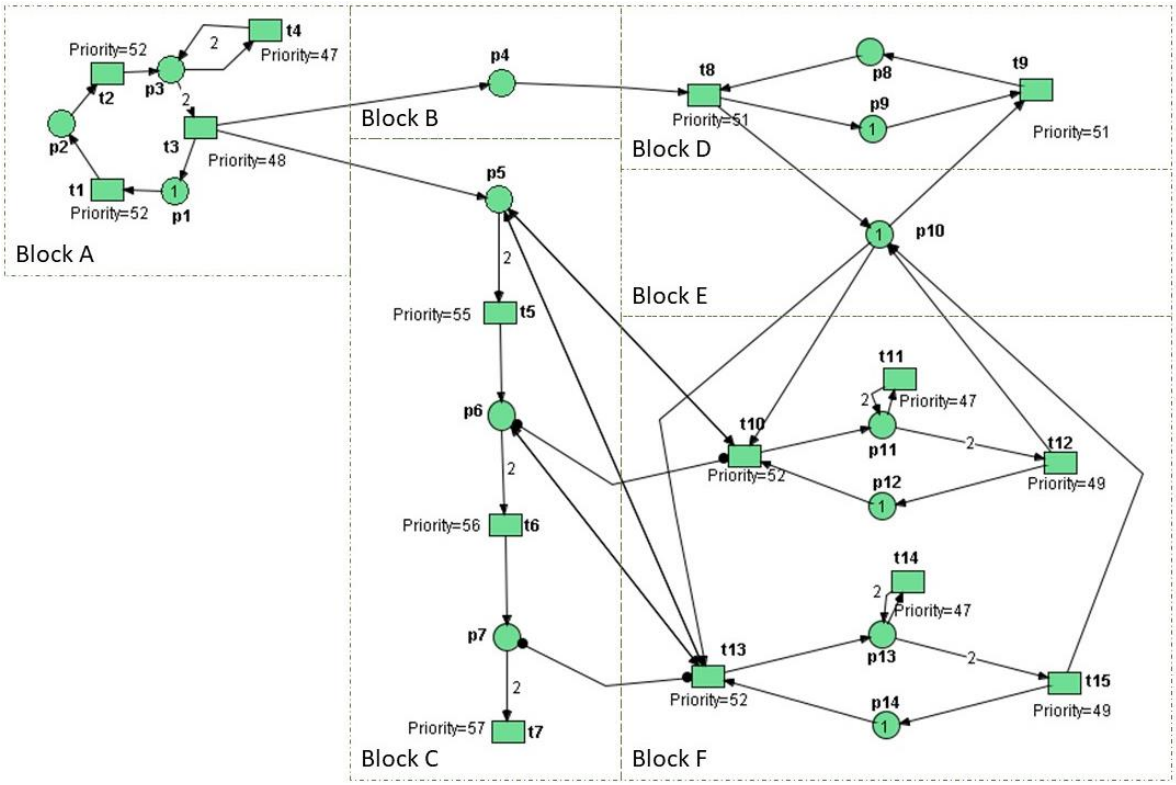

Figure 1: XPN model considering inhibitor arcs and priorities.

$$B_{in} = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&2&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&2&0&0&0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&2&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&2&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&2&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&2 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \end{pmatrix} \quad B_{out} = \begin{pmatrix} 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&2&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&1&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1 \\ 0&0&0&0&0&0&0&0&0&1&2&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&2&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

$$B_h(i,j) = \begin{cases} 1, & \text{for } (i,j) = (6,10) \text{ and } (7,13); \\ 0, & \text{otherwise.} \end{cases}$$

Figure 2: Input, Output, and Inhibitor Matrices for the XPN in Fig. 1.

Pictorially, places in a PN are circles, while transitions are rectangles. Tokens are dots, or numbers, inside places representing a marked place. Input and output flow functions are the weighted arcs from places to transitions and from transitions to places. Matrices $B_{in}[i,j] = I(p_i, t_j)$ and $B_{out}[i,j] = O(p_i, t_j)$ represent input and output flow functions, respectively. A line with an initial small circle that goes from places to transitions denotes the inhibitor function. Matrix $B_h[i,j] = \phi(p_i, t_j)$ represents the inhibitor function. Incidence matrix $B = B_{out} - B_{in}$ and Inhibitor matrix B_h define the structure of a PN.

The Marking function $M(k) \rightarrow \mathrm{W}^m$, denoted as M_k, represents the state of the net at the k^{th} step indicating the number of tokens in every place. Special marking M_0 represents the initial state of the net. Index k increases on every event occurrence to provide a *time basis*. For simplicity, we use $M_k(i)$ to represent the number of tokens on place p_i at event k.

Fig. 1 shows an XPN model representing the behaviour of a simple monotonic rate scheduler for two tasks. According to the model definition, $P = \{p_1, p_2, \ldots, p_{16}\}$, $T = \{t_1, t_2, \ldots, t_{16}\}$, $\rho = \{(t_1, 52), (t_2, 52), (t_3, 48), (t_4, 47), (t_5, 55), (t_6, 56), (t_7, 57), (t_8, 51), (t_9, 51), (t_{10}, 52), (t_{11}, 47), (t_{12}, 49), (t_{13}, 52), (t_{14}, 47), (t_{15}, 49)\}$, and $M_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1]^T$. Functions I, O, ϕ are represented as B_{in}, B_{out}, B_h matrices, shown in Fig. 2. In the XPN in Fig. 1, segments A, B, C, D, E, and F show the submodels for a timer, an OS tick flag, a binary counter, background task, the CPU, and two main tasks, respectively. Notice that considering the sole structure of a PN model without priorities or inhibitor arcs, the net dynamic describes a behaviour, which is more complex than the desired one for a model representing a static monotonic rate scheduling. This is due to the nondeterminism of enabled transitions at the same step k. For example, considering the initial marking M_0 in the XPN in Fig. 1 both transitions t_1 and t_9 are enabled, where anyone could be fired in the case there is no priority information. As we saw in the introduction section, some modelling and simulation tools provide different techniques to solve this conflict, such as probability or manual selection of the transition to fire. In both cases, the conflict is solved, nevertheless, the model's behaviour is still nondeterministic.

There are two main cases to consider in this work. First, to solve the nondeterminism of enabled transitions at the same step k. Second, to solve the enabling of transitions when a condition does not hold, this is, by the absence of a mark in a place. The presented XPN models can address both cases by using priorities of transitions and inhibitor arcs. In addition, we provide all the information to model and simulate XPN models using generic PNML file compatible editors and Matlab/Simulink.

Before introducing the dynamics of XPN models, consider the dynamic of the Place/Transition Petri net as described by the following equation:

$$M_{k+1} = M_k + Bu_k \tag{1}$$

where vector M_k is the state of the system at time k; the Parikh vector $u_k \in \mathrm{W}^n$ represents the enabled transition at marking M_k; B is the incidence matrix; M_{k+1} is the next reached state.

From the initial marking M_0, we should find the enabled transitions considering that one marking may enable one or more transitions for firing. Transition firings produce a token evolution. A transition $t_j \in T$ is enabled at M_k, denoted $[M_k\rangle t_j$, if it fulfils that:

$$M_k \geq B_{in} < t_j > \tag{2}$$

where $< t_j > \in \mathrm{W}^n$ with 1 in the j^{th} position and 0 otherwise.

Therefore, for a transition $t_j \in T$ to be fired using Eq. (1), t_j must be enabled, i.e., holding Eq. (2), and, since the marking M_k may enable more than one transition, i.e., there is nondeterminism, we need one policy to decide which one of them must be fired first.

As mentioned before, the dynamic of an XPN differs from those of Place/Transition Petri nets. In the following section, it is presented the new dynamic induced using inhibitor arcs and priorities of transitions.

See [2, 31, 32] for more information on PN theory.

2.2 eXtended Petri nets dynamics

The marking evolution for XPN uses Eq. (1); nevertheless, to compute the set of enabled transitions from the current net marking M_k, a transition must fulfil Eq. (2) and, additionally, the transition should not be inhibited from the marking M_k holding that:

$$M_k(B_h < t_j >) = \mathbf{0} \tag{3}$$

this is, $M_k(i) \cdot \phi(p_i, t_j) = 0$, $\forall i \in \{1, \ldots, m\}$. In other words, if Eq. (2) holds, then no transition is inhibited by marking M_k, otherwise, transition t_j is inhibited and it cannot be fired. For a set of enabled transitions, e.g., $\{t_i, t_j, \ldots, t_r\}$, at M_k, we use the notation $[M_k\rangle\{t_i, t_j, \ldots, t_r\}$.

After computing the set of enabled transitions, we should consider the priorities of transitions to decide which one should be fired. The following algorithm describes the complete process to compute the dynamics of an XPN model considering Eqs. (1) to (3) and the priority function.

Algorithm 1. ***Compute XPN dynamics with priorities***

Let (G, M_0) be the XPN to compute the dynamic with M_0 as initial marking, B the incidence matrix, ϕ the inhibitor function and ρ the priority function over T.

While continuing the dynamic of the XPN, compute the following:

Step 1. For M_k, compute the partially ordered set $\{T\}_k = \{t_p, t_q, \ldots, t_u\}$ of all enabled transitions from M_k, i.e, $t_i \in \{T\}_k$ if and only if Eq. (2) and Eq. (3) hold for M_k and t_i, ordered by ρ from highest priority to lowest priority, i.e., $\rho(t_p) \geq \rho(t_q) \geq \cdots \geq \rho(t_u)$.

Step 2. Iterate for every $t_j \in \{T\}_k$ in strict order, until $\{T\}_k = \varnothing$:

a) if Eq. (2) and Eq. (3) hold for M_k and t_j, i.e., $[M_k\rangle t_j$, then:
   i) Compute next marking M_{k+1} by using Eq. (1) with $u_k = <t_j>$
   ii) Make $M_k \leftarrow M_{k+1}$

b) Update $\{T\}_k$:
   i) Compute $\{T\}_{k+1} = \{T\}_k - \{t_j\}$
   ii) Make $\{T\}_k = \{T\}_{k+1}$

To show how the XPN dynamic is impacted by inhibitor arcs and by priorities, let us study the next example by following Algorithm 1. Let M_0 be the initial marking for XPN in Fig. 1 such that t_1 and t_9 are the only two enabled transitions, i.e., $[M_0\rangle\{t_1, t_9\}$. According to priority function, $\rho(t_1) > \rho(t_9)$, then $\{T\}_0 = \{t_1, t_9\}$ by Step 1. Thus, t_1 fires by applying Eq. (1) to reach a new marking $M_1 = M_0 + B<t_1> = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1]^T$ by Step 2 a, then $\{T\}_1 = \{T\}_0 - \{t_1\} = \{t_9\}$ by Step 2 b. Since $[M_1\rangle t_9$, then t_9 fires to reach a new marking M_2 by Step 2 a, and now$\{T\}_2 = \{T\}_1 - \{t_9\} = \varnothing$ by Step 2 b.

At this point, the iterative condition in Step 2 breaks, and we continue with Step 1 where $[M_2\rangle t_2$ and, then $\{T\}_2 = \{t_2\}$. Now, t_2 fires to reach marking M_3 by Step 2 a, and $\{T\}_3 = \varnothing$ by Step 2 b. Iterative condition in Step 2 breaks, then we continue with Step 1 where $[M_3\rangle t_4$, and $\{T\}_3 = \{t_4\}$. Transition t_4 models the counting of a Timer that fires repeatedly until p_3 holds s tokens, i.e, the timer overflows (with s number of intermediate markings, $s = 2$ in this example), to end in a new marking, say M_4, such that $M_4\rangle\{t_3, t_4\}$, then $\{T\}_4 = \{t_3, t_4\}$ by Step 1. Thus, t_3 fires to reach a new marking M_5 by Step 2 a, and $\{T\}_5 = \{t_4\}$ by Step 2 b. Nevertheless, the condition in Step 2 a does not hold since the new marking M_5 disables the firing of t_4. Then, the algorithm continues again at Step 1 from M_5.

Considering $M_5 = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1]^T$. Thus, $\{T\}_5 = \{t_1, t_8\}$, since $[M_5\rangle\{t_1, t_8\}$ and $\rho(t_1) > \rho(t_8)$, i.e., marking M_5 enables the firing of both t_1 and t_8, and the priority of t_1 is higher than that of t_8. In this case, t_8 fires to reach a new marking M_6, and $\{T\}_6 = \{t_1\}$. Since $[M_6\rangle t_1$, then t_1 fires to reach $M_7 = [0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1]^T$. At this marking, transitions t_2, t_9 and t_{10} are enabled, i.e., $[M_7\rangle\{t_2, t_{10}, t_9\}$.

Notice that t_2 and t_{10} have the same priority and t_9 has the least priority among the three, i.e., $\rho(t_2) = \rho(t_{10}) > \rho(t_9)$. Thus, $\{T\}_7 = \{t_2, t_{10}, t_9\} = \{t_{10}, -t_2, -t_9\}$. In this case, the simulation process fires t_2 and t_{10} in any order. Perhaps, if the system requires the firing of t_2 and t_{10} to be deterministic, the priority of the one that must be fired first should be greater, e.g., by assigning $\rho(t_2) = 54$.

Observe that firing t_{10} disables the firing of t_9, which is the desired behaviour for the scheduler, i.e., the execution of main tasks has priority over background tasks. The firing of t_3 causes the interrupt service routine (ISR) to execute, which increments the Counter (p_3) and

turns on the OS tick flag (p_4). When the OS Tick flag is true (p_4), the scheduler checks if one of the main tasks is ready to start (firing t_{10} or t_{13}). If such a case, the scheduler runs the activated task (either p_{11} or p_{13}). Otherwise, the background task (p_9) returns to its running state (p_8).

By comparing the dynamics of both XPN and the Petri net without inhibitor arcs and priorities of transitions, we can notice that only XPN is capable of being deterministic. This is an important property for our example system as well as for many complex systems.

3. XPN FROM PNML TO MATLAB DATA

The modelling and simulation process for XPN models is outlined in Fig. 3. In the following sections, the complete process is presented in detail.

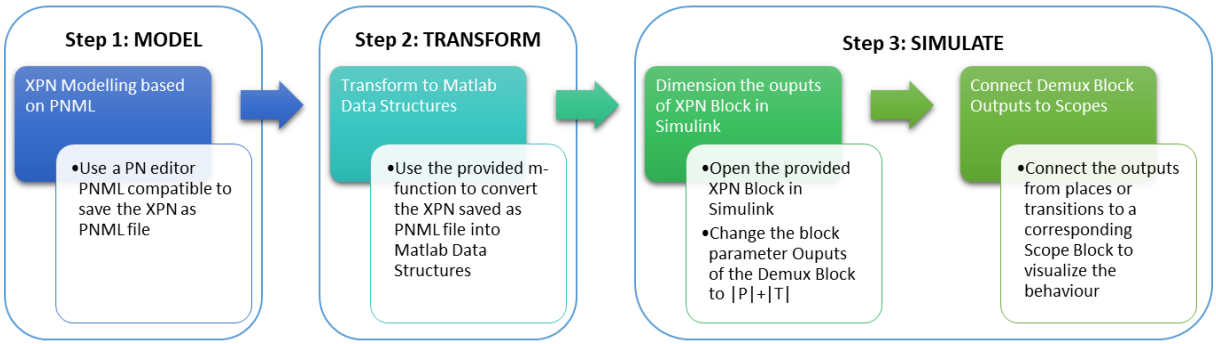

Figure 3: Modelling and simulation process of XPN.

3.1 XPN model as PNML

PNML (Petri Net Marking Language) is an XML-based format defined by the standard ISO/IEC 15909 Part 2 [33]. Implementing this standard allows one to model PN in one editor software or tool, save it as a PNML format, and make it compatible with other software applications or tools based on this format.

In this work, we use the tool called renew as a PN editor [34]. It is based on Java and is easy to install and use for our modelling purposes. The PN editor does not allow including priority information, nevertheless, we can include it as an annotation for the transitions, and then export the PN model as a PNML file. For further information about this tool, the authors refer to [34].

Fig. 1 shows an example of the PN model using renew and including the priority information as an annotation for transitions. These annotations appear as text tags for transitions on the PNML file.

3.2 Transform from PNML to MATLAB data structures

Once the Petri net model is saved as a PNML file then it is transformed into MATLAB data structures to exploit the functionalities provided by MATLAB. We can implement m-functions to analyse the net, use algebra in the command line, or use any other MATLAB functionality. Moreover, we can use Simulink for simulation purposes as explained later.

The herein proposed m-function to extract XPN data structures from the PNML file is named *MfromPNML* with the name of the PNML file (**.pnml*) as input and B_{in}, B_{out}, B_h, M_0, *Prio*, and *I* as outputs. *Prio* is a transposed matrix-vector representing the priority function ρ for every *n* transition in the XPN, and *I* is a transposed, unitary, matrix-vector of size *n* created to help in the XPN block simulation, as we will show it later. The first step of the *m* function uses the *xml2struct.m* function published by Falkena [35]. In the conclusion section, there is a link to download the XPN example as a PNML file and the *m* function MfromPNML as an M file.

4. XPN SIMULATION RESULTS

4.1 Dynamics of the XPN Matlab block

Fig. 4 a shows an overview of the implemented blocks in Simulink. The simulation model shows one input (*I*) and three outputs (*State*, *Parallel Transitions*, *Sequence of Fired Transitions*). *I* is a unitary, matrix-vector of size *n* to support the selection of the places for PN driven by an input signal (not used in this work). The state is the marking on every place at every simulation step. We use a Demux block to separate the $m = 14$ outputs for the example here presented, one output for each place, and then we connect each one to a scope block that shows the behaviour of the corresponding place.

Fig. 4 b shows the composed Block for an XPN Block. Input 1 to XPN Block is not used in this work, but it is useful for nets driven by input signals, such as interpreted Petri nets shown in [16]. The main block is the XPN, which is based on an M file *S* function named *my_xpn.m* that implements the dynamic of the net. The link for the files of the Simulink model named *XPN_Simulink_Model.slx*, and the required *m* function, is provided in the Conclusion section.

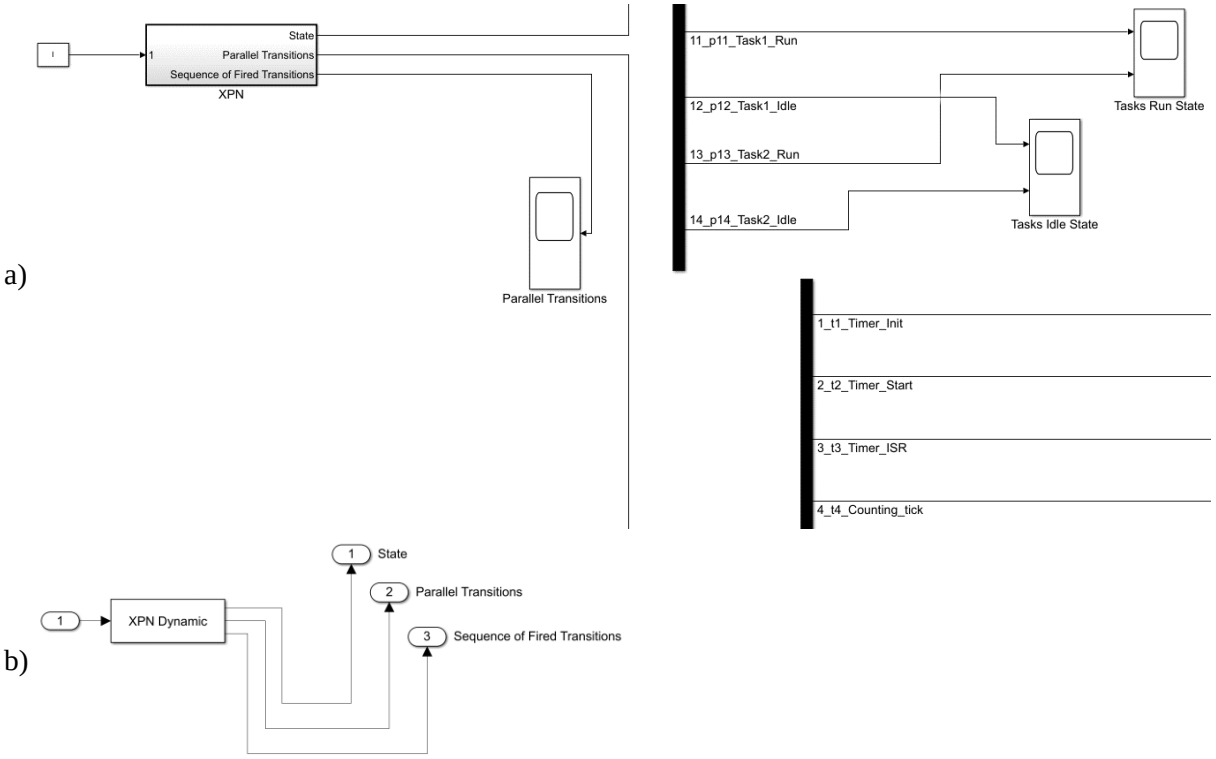

Figure 4: Simulink framework overview: a) Partial blocks configuration, b) XPN Dynamic block.

4.2 Simulation example

The outputs obtained after simulating the *XPN_Simulink_Model* for 100 discrete steps for the counter status, the tasks activation and running, and CPU idle periods are shown in Fig. 5 a, Fig. 5 b, and Fig. 5 c, respectively. Notice that every place in Fig. 1 Block C is a digit, i.e., p_5 is the Least Significant Bit (LSB) and p_7 is the Most Significant Bit (MSB). Every ISR from the timer (t_3) increments the counter by adding one mark in the LSB (p_5) of the binary counter. Observe in Fig. 5 a that p_5 changes immediately to 0 right after it gets 2 marks, which makes p_6 change to 1, and the same behaviour maintains from p_6 to p_7.

Observe in Fig. 5 b that Task 1 activates every time that the counter has 1 mark in place p_5, 0 marks in place p_6, and either 0 or 1 mark in p_7, as pointed out by the red dashed vertical line; and Task 2 activates every time that the counter has 1 mark in place p_5, 1 mark in place p_6, and

0 marks in p_7, as pointed out by the orange long-dash vertical line. This activation policy assures a rate of monotonic scheduling for task 1 and task 2, represented by places p_{11} and p_{13} in block F, in the same figure. Correspondingly, the start of CPU usage is marked just when a task is activated, as shown in Fig. 5 c, and pointed out by the vertical lines.

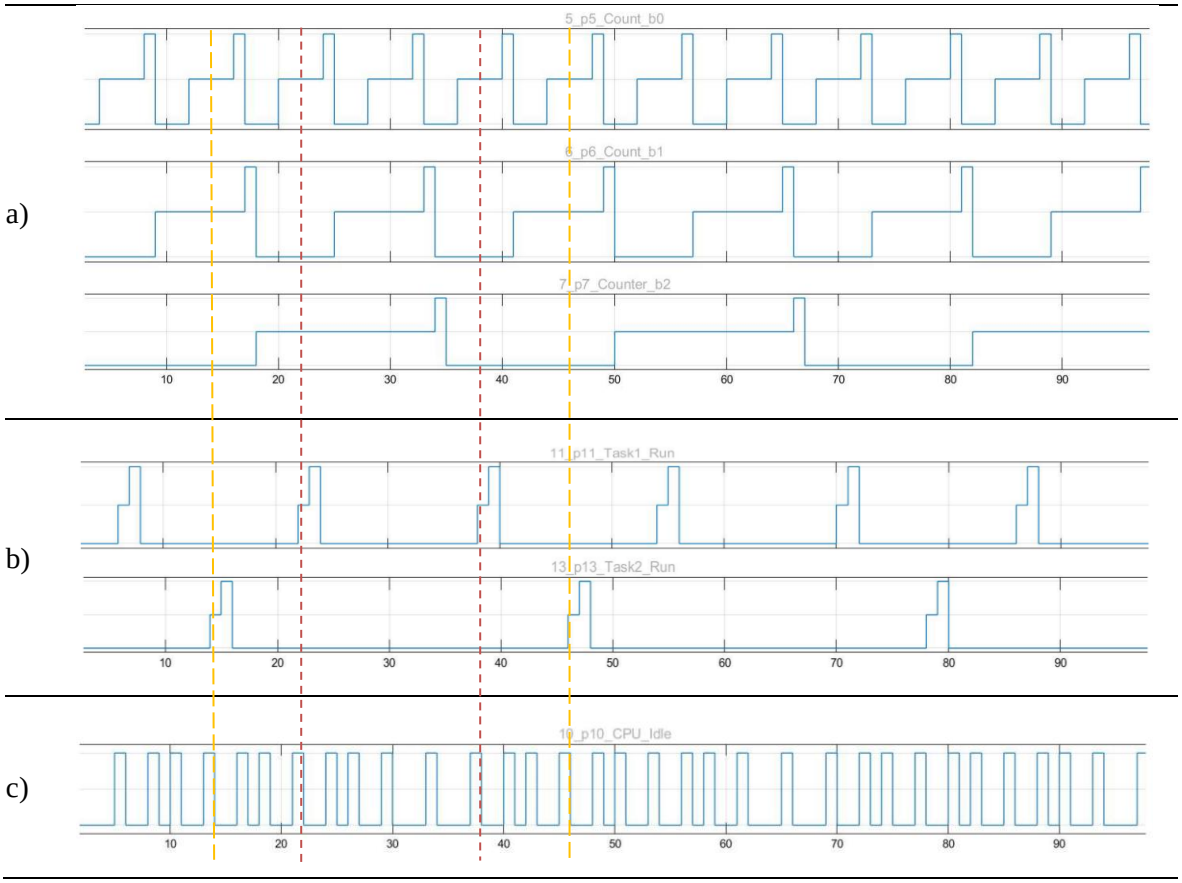

Figure 5: XPN Simulation outputs to visualize: a) the binary counting status, b) the activation and running period of tasks, and c) the CPU idle periods.

The analysis of logic correctness as well as temporal analysis is one of the main advantages that brings the herein proposed methodology of modelling and simulation of XPN models using Matlab/Simulink. The analysis of output results from Fig. 5 a allows us to confirm that the counter is binary counting correctly and to validate that every task is activated and running at the time (step) we expect, as shown in Fig. 5 b. Additionally, with an appropriate analysis of the outputs shown in Fig. 5 c, we can find the CPU usage as well.

6. CONCLUSION

This work presented a methodology to work with Petri nets that includes inhibitor arcs and priority functions by using Matlab data structures and Simulink. One of the main objectives to work with priorities on transitions is to address the nondeterminism induced when there is more than one enabled transition from one single marking or state of the system net. Additionally, working with inhibitor arcs eases the modelling by allowing us to block some firings from unmarked places. First, we presented a definition for a type of extension of Petri nets models, named XPN, which considers prioritization on transitions, and inhibitor and bidirectional arcs, to model and simulate complex systems. Second, we provided equations and the algorithms for the dynamics of the XPN induced by inhibitor arcs and prioritization of transitions. Then, we introduced the m-functions to transform XPN from PNML to Matlab data types and for the

dynamics of the XPN. Later, we presented the Simulink block configuration to perform the simulation of XPN by using the XPN dynamics algorithm herein presented. Finally, we showed an exploratory case study Rate Monotonic Scheduling for two tasks with harmonic periods to show the complete modelling and simulation process, as well as to revise some simulation results that we can obtain by using the Simulink block configuration presented. The simulation process can be extended to other types of Petri nets. The PNML file of the XPN example model, the *m* functions and the Simulink model presented in this work are found in *https://la.mathworks.com/matlabcentral/fileexchange/123840-simulation-of-xpn-models*.

ACKNOWLEDGEMENT

The authors thank the Department of Electronics, Systems, and Informatics of ITESO University for the financial support of this project.

REFERENCES

[1] Petri, C. A. (1966). *Communication with Automata,* Research and Technology Division, Rome Air Development Center, Rome (USA)

[2] Desel, J.; Reisig, W. (1996). Place/transition Petri nets, Reisig, W.; Rozenberg, G. (Eds.), *Advanced Course on Petri Nets*, Springer, Berlin, 122-173, doi:10.1007/3-540-65306-6_15

[3] Gomes, L.; Barros, J. P.; Costa, A.; Nunes, R. (2007). The input-output place-transition Petri net class and associated tools, *2007 5th IEEE International Conference on Industrial Informatics*, 509-514, doi:10.1109/INDIN.2007.4384809

[4] Ramchandani, C. (1973). *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*, PhD Thesis, Massachusetts Institute of Technology, Cambridge

[5] Cerone, A.; Maggiolo-Schettini, A. (1999). Time-based expressivity of time Petri nets for system specification, *Theoretical Computer Science*, Vol. 216, No. 1-2, 1-53, doi:10.1016/S0304-3975(98)00008-5

[6] Alla, H.; David, R. (1998). A modelling and analysis tool for discrete events systems: continuous Petri net, *Performance Evaluation*, Vol. 33, No. 3, 175-199, doi:10.1016/S0166-5316(98)00016-9

[7] David, R.; Alla, H. (2001). On hybrid Petri nets, *Discrete Event Dynamic Systems*, Vol. 11, No. 1-2, 9-40, doi:10.1023/A:1008330914786

[8] Simon, E.; Oyekan, J.; Hutabarat, W.; Tiwari, A.; Turner, C. J. (2018). Adapting Petri nets to DES: stochastic modelling of manufacturing systems, *International Journal of Simulation Modelling*, Vol. 17, No. 1, 5-17, doi:10.2507/IJSIMM17(1)403

[9] Xu, S. Z. (2019). A Petri net-based hybrid heuristic scheduling algorithm for flexible manufacturing system, *International Journal of Simulation Modelling*, Vol. 18, No. 2, 325-334, doi:10.2507/IJSIMM18(2)CO6

[10] Reinhardt, K. (2008). Reachability in Petri nets with inhibitor arcs, *Electronic Notes in Theoretical Computer Science*, Vol. 223, 239-264, doi:10.1016/j.entcs.2008.12.042

[11] Westergaard, M.; Verbeek, H. M. W. (2011). Efficient implementation of prioritized transitions for high-level Petri nets, *Proceedings of the 2011 Petri Nets and Software Engineering*, 27-41

[12] Padberg, J. (2015). Reconfigurable Petri nets with transition priorities and inhibitor arcs, Parisi-Presicce, F.; Westfechtel, B. (Eds.), *Graph Transformation*, Springer, Cham, 104-120, doi:10.1007/978-3-319-21145-9_7

[13] Campos-Rodríguez, R.; Alcaraz-Mejía, M.; Mireles-García, J. (2007). Supervisory control of discrete event systems using observers, *2007 Mediterranean Conference on Control & Automation*, 7 pages, doi:10.1109/MED.2007.4433816

[14] Campos-Rodriguez, R.; Alcaraz-Mejia, M. (2010). A Matlab/simulink framework for the design of controllers and observers for discrete-event systems, *Elektronika ir Elektrotechnika*, Vol. 99, No. 3, 63-68

[15] Wisniewski, R.; Bazydlo, G.; Szczesniak, P.; Wojnakowski, M. (2019). Petri net-based specification of cyber-physical systems oriented to control direct matrix converters with space vector modulation, *IEEE Access*, Vol. 7, 23407-23420, doi:10.1109/ACCESS.2019.2899316

[16] Alcaraz-Mejia, M.; Lopez-Mellado, E.; Ramírez-Treviño, A.; Rivera-Rangel, I. (2003). Petri net based fault diagnosis of discrete event systems, *Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics*, 4730-4735, doi:10.1109/ICSMC.2003.1245731
[17] Lefebvre, D. (2006). Sensoring and diagnosis of DES with Petri net models, *IFAC Proceedings Volumes*, Vol. 39, No. 13, 1145-1150, doi:10.3182/20060829-4-CN-2909.00191
[18] Alcaraz-Mejia, M.; Lopez-Mellado, E. (2006). Petri net model reconfiguration of discrete manufacturing systems, *IFAC Proceedings Volumes*, Vol. 39, No. 3, 547-552, doi:10.3182/20060517-3-FR-2903.00283
[19] Alcaraz-Mejia, M. I.; Campos-Rodriguez, R.; Lopez-Mellado, E.; Ramirez-Trevino, A. (2015). Partial reconfiguration of control systems using Petri nets structural redundancy, *Information Technology and Control*, Vol. 44, No. 3, 287-301, doi:10.5755/j01.itc.44.3.8783
[20] Alcaraz-Mejia, M.; Campos-Rodriguez, R. (2019). A framework based on Matlab/Simulink for the simulation of DES using Petri net models, *International Journal of Simulation Modelling*, Vol. 18, No. 3, 420-431, doi:10.2507/IJSIMM18(3)479
[21] Huang, Z.; Yang, J. J. (2022). A new model for optimization of cell scheduling considering inter-cell movement, *International Journal of Simulation Modelling*, Vol. 21, No. 1, 136-147, doi:10.2507/IJSIMM21-1-CO1
[22] Wang, S. R.; Huang, Q. (2022). A hybrid code genetic algorithm for VRP in public-private emergency collaborations, *International Journal of Simulation Modelling*, Vol. 21, No. 1, 124-135, doi:10.2507/IJSIMM21-1-595
[23] Julvez, J.; Matcovschi, M. H.; Pastravanu, O. (2014). MATLAB tools for the analysis of Petri net models, *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation*, 12 pages, doi:10.1109/ETFA.2014.7005053
[24] Matcovschi, M.; Popescu, C.; Pastravanu, O. (2006). A new approach to hybrid system simulation: development of a Simulink library for Petri net models, *Control Engineering and Applied Informatics*, Vol. 7, No. 4, 55-62
[25] Matcovschi, M. H.; Mahulea, C.; Lefter, C.; Pastravanu, O. (2006). Petri net toolbox in control engineering education, *2006 IEEE Conference on Computer Aided Control Systems Design*, 2298-2303, doi:10.1109/CACSD-CCA-ISIC.2006.4776998
[26] Júlvez, J.; Mahulea, C.; Vázquez, C. R. (2012). SimHPN: a MATLAB toolbox for simulation, analysis and design with hybrid Petri nets, *Nonlinear Analysis: Hybrid Systems*, Vol. 6, No. 2, 806-817, doi:10.1016/j.nahs.2011.10.001
[27] Sessego, F.; Giua, A.; Seatzu, C. (2008). HYPENS: a Matlab tool for timed discrete, continuous and hybrid Petri nets, van Hee, K. M.; Valk, R. (Eds.), *Applications and Theory of Petri Nets*, Springer, Berlin, 419-428, doi:10.1007/978-3-540-68746-7_28
[28] Lomazova, I. A.; Popova-Zeugmann, L. (2016). Controlling Petri net behavior using priorities for transitions, *Fundamenta Informaticae*, Vol. 143, No. 1-2, 101-112, doi:10.3233/FI-2016-1306
[29] Alhazov, A.; Ivanov, S.; Pelz, E.; Verlan, S. (2016). Small universal deterministic Petri nets with inhibitor arcs, *Journal of Automata, Languages, and Combinatorics*, Vol. 21, No. 1-2, 7-26, doi:10.25596/jalc-2016-007
[30] Reinhardt, K. (2008). Reachability in Petri nets with inhibitor arcs, *Electronic Notes in Theoretical Computer Science*, Vol. 223, 239-264, doi:10.1016/j.entcs.2008.12.042
[31] Murata, T. (1989). Petri nets: properties, analysis and applications, *Proceedings of the IEEE*, Vol. 77, No. 4, 541-580, doi:10.1109/5.24143
[32] Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*, 1st edition, Prentice Hall, Hoboken
[33] ISO/IEC 15909-2:2011(en). Systems and Software Engineering – High-level Petri Nets – Part 2: Transfer Format, from *https://www.iso.org/obp/ui/es/#iso:std:iso-iec:15909:-2:ed-1:v1:en*, accessed on 15-07-2022
[34] Theoretical Foundations Group of the Department for Informatics of the University of Hamburg. Renew: The Reference Net Workshop, from *http://www.renew.de/*, accessed on 04-28-2022
[35] Wouter Falkena. Xml2struct, from *https://www.mathworks.com/matlabcentral/fileexchange/28518-xml2struct*, accessed on 27-06-2021